HATE SPEECH DETECTION USING DEEP LEARNING

B.Tech. Project Final Report

B.Tech.

by

Anurag Srivastava (2018IMT-019)

Under the supervision of

Dr. Vinal Patel and Dr. Santosh Singh Rathore

विश्वजीवनामृतं ज्ञानम्

ABV INDIAN INSTITUTE OF INFORMATION TECHNOLOGY AND MANAGEMENT GWALIOR-474 010

2021

CANDIDATES DECLARATION

I hereby certify that the work, which is being presented in the report, entitled **Hate Speech Detection using Deep Learning**, in partial fulfillment of the requirement for the award of the Degree of **Bachelor of Technology** and submitted to the institution is an authentic record of our own work carried out during the period *June 2021* to *october 2021* under the supervision of **Dr. Vinal Patel** and **Dr. Santosh Singh Rathore**. We also cited the reference about the text(s)/figure(s)/table(s) from where they have been taken.

[illegible]

31/10/2021

Date: Signatures of the Candidates

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

[illegible] [illegible]

Date: Signatures of the Research Supervisors

ABSTRACT

Hate speech has been a big problem in today's internet age. Unchecked content like in Twitter is frequently being consumed by the users. Social distress and misconceptions are the one of the most dangerous outcomes of hate speech. In this report we explore various methods to detect hate speech using deep learning techniques. This endeavour is difficult due of the intricacy of natural language constructs. In this work, we will use kaggle data set to classify text as hate speech or not. We will go through machine learning, natural language processing, Deep learning methods like ANN, CNN, RNN and LSTM. Due to sequential nature of text data RNN turns out to be most effective method.

Keywords: Sliding Window · Hate Speech · Bag-Of-Words · Convolutional NN · Deep Learning · NLP · Word Embedding · Offensive Content · RNN · LSTM

ACKNOWLEDGEMENTS

I am highly indebted to **Dr. Vinal Patel** and **Dr. Santosh Singh Rathore**, and obliged for giving me the autonomy of functioning and experimenting with ideas. I would like to take this opportunity to express my profound gratitude to them not only for their academic guidance but also for their personal interest in my project and constant support coupled with confidence boosting and motivating sessions which proved very fruitful and were instrumental in infusing self-assurance and trust within me. The nurturing and blossoming of the present work is mainly due to their valuable guidance, suggestions, astute judgment, constructive criticism and an eye for perfection. My mentor always answered myriad of my doubts with smiling graciousness and prodigious patience, never letting me feel that I am novices by always lending an ear to my views, appreciating and improving them and by giving me a free hand in our project. It's only because of their overwhelming interest and helpful attitude, the present work has attained the stage it has.

Finally, I am grateful to our Institution and colleagues whose constant encouragement served to renew my spirit, refocus our attention and energy and helped my in carrying out this work.

Anurag

31/10/2021

(Anurag Srivastava)

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION AND RELATED WORK

This chapter gives brief introduction of the project as well as literature survey and related works done over the task.

1.1 INTRODUCTION

1.1.1 Hate Speech Detection

Hate Speech is an immediate or roundabout assertion focused on an individual or gathering of individuals proposed to belittle and mistreat another or utilize deprecatory language dependent on nationality, religion, incapacity, sex, or sexual direction. Hate speech has continuously increased as a result of the tremendous increase in user-developed content on social media. Hate speech directed towards a specific person or group of individuals can result in personal anguish, cyberbullying, societal panic, and discrimination. In reaction to the rise in hate speech on social media, a few of studies on automatic hate speech detection have been published in an attempt to reduce online harassment. In this work, we will go through various methods and choose the best possible method that gives us the best accuracy.

1.2 MOTIVATION

In a study conducted in USA [link] , while utilising social media, 12% of students said they frequently saw racist hate speech. Overall, 52% of respondents said they saw racist hate speech on social media on a regular or irregular basis. With the increase in data all over, most of which remains unchecked and unregulated, hate speech is quite prevalent. The field of hate speech detection remains inconclusive and various ongoing research is still going on. This is the motivation to explore various available tools to detect hate speech in this project.

1.3 LITERATURE REVIEW

Significant early work on hate-speech recognition was done by Spertus (1997), who constructed a model framework Smokey utilizing a C4.5 choice tree generator to decide highlight-based guidelines that could order harmful messages. From that point forward, disdain discourse identification has accomplished achievements, and a few models have been prepared to recognize disdain discourse. Yin et al. (2009) were quick to utilize a regulated learning way to deal with distinguish provocation on web 2.0. They grouped online media posts utilizing a support-vector machine (SVM) in light of nearby relevant and assessment features. Zampieri and Malmasi (2017) [1] inspected character n-grams, word n-grams and skip-grams to recognize disdain discourse in web-based media. They prepared their classifier on an English informational index with three marks and accomplished an exactness of 78%.

Hate speech detection on web platforms has been made possible because to advances in ML and NLP. ML and Deep Learning approaches for automated-hate-speech and offensive material detection have been the subject of numerous scientific studies. Character level and word level n-grams are the commonly used features in ML based approaches, and so on Albeit directed ML based methodologies have utilized diverse content mining-based components like surface provisions, assumption examination, lexical resources,linguistic highlights, information based elements, or client based and stage based metadata, they require a distinct element extraction approach. These days, the neural-network models apply text representation and deep learning approaches such as CNN [2], Bi-directional Long-Short-Term Memory Networks (LSTMs) [3], and Bidirectional Encoder Representations from Transformers [4] to improve the performance of hate speech detection models.

1.4 OBJECTIVE

The main objective is to design a deep learning model that is capable of classifying given text as hate speech or not. This is essentially a binary classification problem. We concentrate on Kaggle dataset, which is the most extensively utilised data source in the study of abusive language. We use all publicly available datasets with tweets categorised as various sorts of abuse and written in English. We will go through basic ML techniques to Convolutional Neural Networks and finally Recurrent Neural Networks, to see which is the most promising approach to yield the best accuracy.

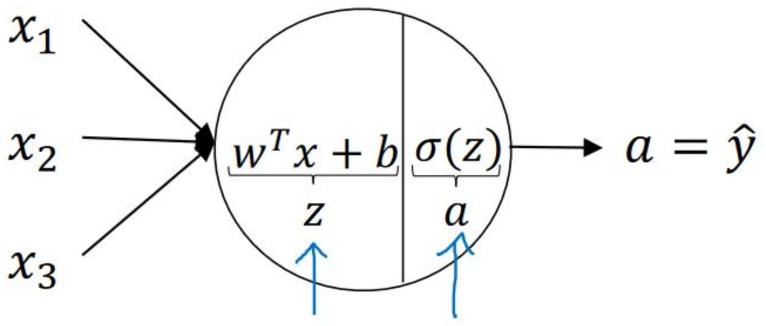
CHAPTER 2

IMPLEMENTATION AND SYSTEM ARCHITECTURE

2.1 SYSTEM ARCHITECTURE

In this section we go through some of the basic architecture of Deep Learning Models and a few of those related closely to the task.

2.1.1 Deep Neural Networks

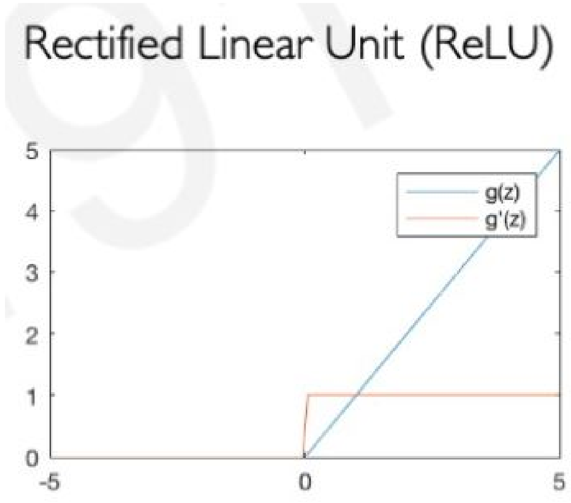


$$z = w^T x + b$$

$$a = \sigma(z)$$

Figure 2.1: Neuron representation.

xi denote input features. *w* denotes weight corresponding to the features and *b* is bias vector to shift preference of features. *a* is Activation Function. Our Model uses ReLU (2.2) activation function in the perceptron. Standard NN is made using a



$$g(z) = \max(0, z)$$

$$g'(z) = \begin{cases} 1, & z > 0 \\ 0, & \text{otherwise} \end{cases}$$

Figure 2.2: ReLU.

hidden layer consisting of inter-related input features and nodes in hidden layer i.e. combination of multiple Neurons linked in a complete graph fashion. Deep Neural Network is extension of Standard NN (shown in 2.3) that contains multiple hidden layers. Summation of difference of output, *y*, from the actual output is called Loss and function that calculates this is using the input features, weights and activation function is called Loss Function. While training a Deep Neural Model multiple iterations are carried out to get the values of *w* so that Loss function is minimal. Based on this very fundamental yet complex ideology we proceed using different techniques to identify best fit model for our task.

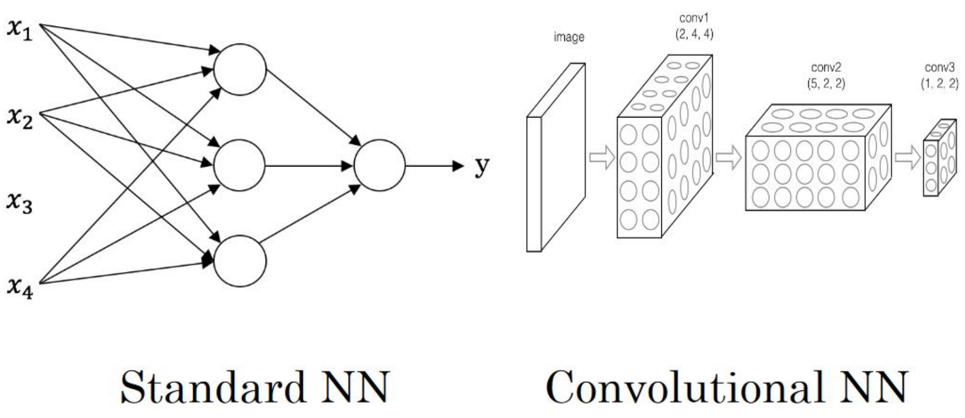


Figure 2.3: Neural Network representation.

2.1.2 Natural Language Processing

Natural language preparing (NLP) is a subject of software engineering—explicitly, a part of man-made consciousness (AI)— concerning the capacity of PCs to get text and verbally expressed words in the very way that people can. Computational semantics—rule-based human language demonstrating—is joined with factual, AI, and profound learning models in NLP. These advancements, when utilized together, permit PCs to deal with human language as text or discourse information and 'comprehend' its full significance, including the speaker or author's purpose and notion. A few NLP exercises assist the machine with getting what it's retaining by separating human content and discourse input in manners that the PC can comprehend. The following are some of these responsibilities:

- Speech recognition
- Part of speech tagging
- Sentiment analysis

Sentiment Analysis is closely related to our task that makes understanding and usage of NLP in the project crucial.

2.1.3 Architecture-Flow

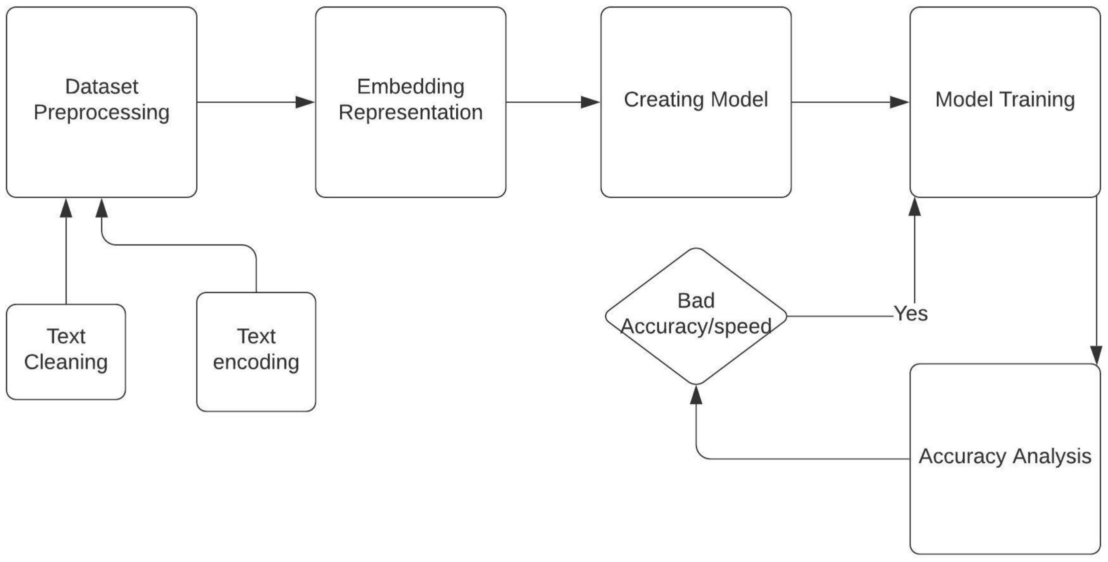


Figure 2.4: System-Architecture

In the diagram above, the sequential order of process is demonstrated. The first step is Data Pre-processing, which involves cleaning of text data to remove unwanted data, and Text encoding that converts text into numerical form. Then we proceed to Embedding representation that turns numerical data into matrices so that mathematical operations can be performed. Further we have Model Creation and Training, where we choose which model to choose and number of its iterations. At last we analyze the accuracy and speed of model, and if we don't get good accuracy then we can use some techniques and train the model again. In the next section we will elaborate further on these methodologies.

2.2 METHODOLOGY

2.2.1 PreProcessing Dataset

We will use standard Natural Language Processing (NLP) pre-processing techniques such as Tokenization, Stemming, and Lemmatization. nltk library is videly used for text pre-processing. Below functions are available in the same library too.

2.2.1.1 Stopwords

A sentence may have a lot of words which don't contribute to the decision of hate speech detection, like 'is,and,the', so we use stopwords and remove occurrences of those words. This helps to reduce the overhead of model as number of parameters decrease.

2.2.1.2 Tokenization

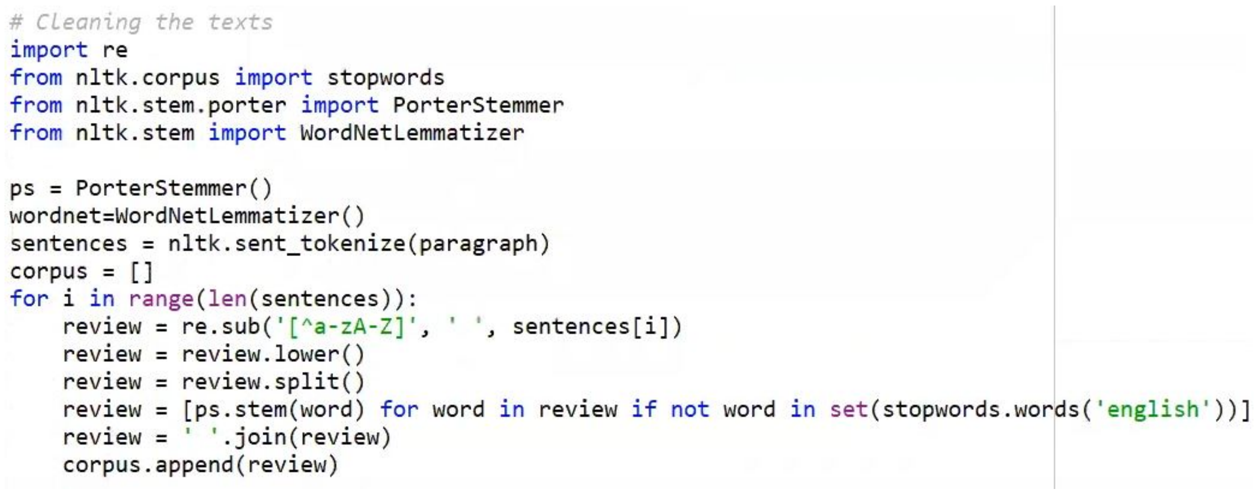
Since, we can not make use of text data in the model because mathematical functions need numerical data, so we tokenize our paragraph/Sentences/Words and assign them a token just like indexing them sequentially to associate text data to numerical form.

2.2.1.3 Stemming

Often, in a document we have different forms of the same word of two words seem to have some common sequence of letters in them. In the stemming technique we select the common prefix from words and use stem word in place of actual words, this reduces the total number of words in the corpus that can help us reduce runtime.

2.2.1.4 Lemmatization

Selecting root word like that in stemming may not yield us a meaningful word, like 'history' and 'historical' both give 'histor' as root word using Stemming, but this doesn't make sense and it maybe possible that to similar spelled words having completely different meaning have same root word. To overcome this problem we use Lemmatization that selects root words based on the semantics rather than simple common prefix. Eg. 'history' and 'historical' will have 'history' as root word. This method also helps us to reduce total number of words in the corpus.



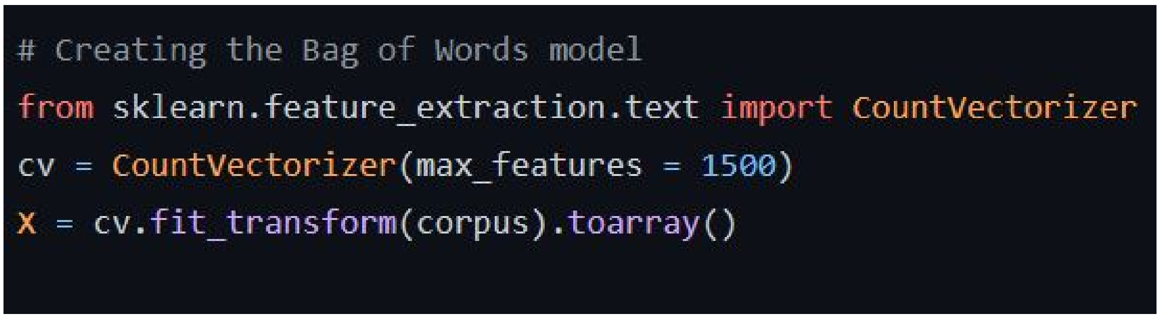
```
# Cleaning the texts
import re
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from nltk.stem import WordNetLemmatizer

ps = PorterStemmer()
wordnet=WordNetLemmatizer()
sentences = nltk.sent_tokenize(paragraph)
corpus = []
for i in range(len(sentences)):
    review = re.sub('[^a-zA-Z]', ' ', sentences[i])
    review = review.lower()
    review = review.split()
    review = [ps.stem(word) for word in review if not word in set(stopwords.words('english'))]
    review = ' '.join(review)
    corpus.append(review)
```

Figure 2.5: Pre-Processing

Code in Figure 2.5 shows usage of nltk library for stemming and removing stopwords. Now we will discuss some of the embedding techniques starting from machine learning and then moving on to deep learning techniques.

2.2.2 Bag Of Words

Bag of Words (BOW) model represents the text in 2-D matrix form, very similar to adjacency matrix of a graph. One one side we have labels denoting sentence index and on the other side we have the words itself. In binary BOW we have data in form of 1 (if word is present in the sentence) or 0. In regular BOW we store the frequency of words in the matrix corresponding to the sentence.



```
# Creating the Bag of Words model
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer(max_features = 1500)
X = cv.fit_transform(corpus).toarray()
```

Figure 2.6: BOW

Figure 2.7 shows the usage of sklearn library that is being used for Bag of Words. The disadvantage of BOW is that it fails to infer semantic meaning from sentences and turns out to give bad and unreliable accuracy.

2.2.2.1 TF-IDF

Term Frequency and Inverse Document Frequency is an improvisation over BOW.

$$TF(word)(i) = \frac{Fi(word)}{Ni}$$

$$IDF(word) = log\frac{N}{F'(word)}$$

After calculating both, we multiply TF matrix with values of IDF for each word. This method adds semantic meaning of the word w.r.t the sentences and also helps in reducing redundant features. This improves the accuracy over BOW but still isn't practically usable.

Similarly other techniques involving machine learning provide very little improvement and none of them can be used practically. In machine learning algorithms, we cannot establish much regarding correlation of the words and sequence of the words. For example, word A and B may be independently not denoting hate speech, but when used together make the sentence likely to be in negative side. Moreover order of the words play a important role in the meaning of the sentences. These issues cannot be covered by machine learning, that is why we move on to the deep learning techniques that are better at feature extraction, though we can use the matrices generated by Word embedding or TF-IDF for further deep neural networks input.

2.2.3 Deep-Learning Approaches

2.2.3.1 Logistic Regression

Hate speech detection is basically a binary classification problem, and the very first approach that comes in mind is logistic regression. Using sigmoid activation function and few hidden layers we are able to acheive accuracy of >70%. Improvement in accuracy is because the hidden layers are able to correlate words to some extent, and able to extract features better than machine learning algorithms.

2.2.3.2 Convolutional Neural Networks

Now we move to one of the best methods for feature extraction in deep learning. We will first pre process our dataset, then use word embedding to map the sequence into a vector, that is we will be using Word embedding technique to create a matrix from the sequence of words. We can use different window sizes for a filter with MxM dimension over the input matrix for feature extraction. Different window sizes in the sliding window ensure selection of bi-gram, tri-gram features on the input matrix, which results in the better feature extraction. Then we apply max-pooling and down-sampling and these techniques significantly improve F1 score. Finally after this we get accuracy >85% which is best among all techniques till now. The main reason is the feature extraction and sliding window, that increases chances of correlated words to form one feature.

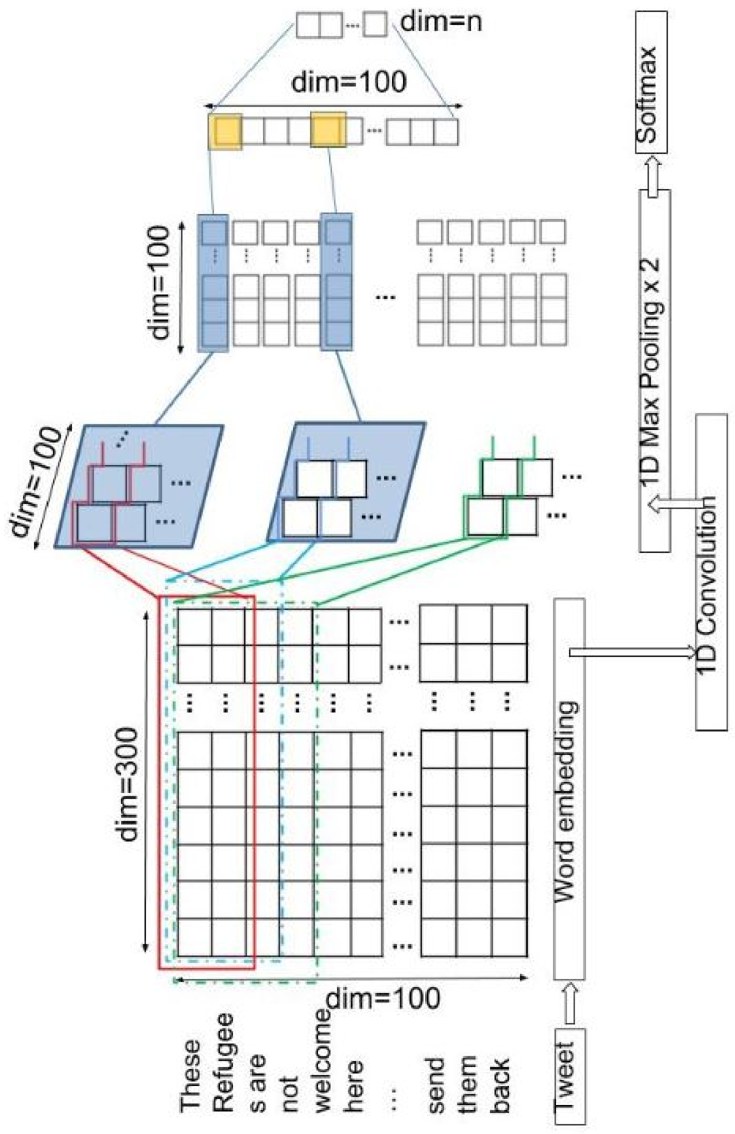


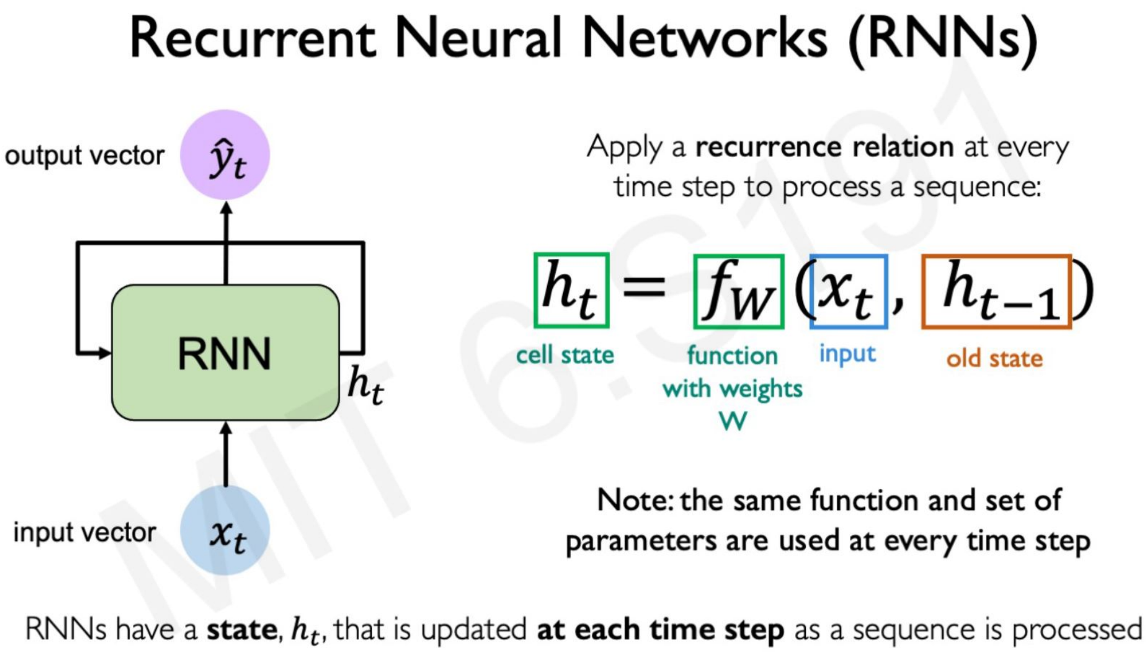
Figure 2.7: CNN

2.2.3.3 RNN + LSTM

Though CNN do good job in feature extraction, it still doesn't make use of sequence of words that well. Therefore we need a model that specifically targets feature extraction using sequence of words in the sentence. Recurrent Neural Networks are used whenever there is a sequential data.

2.3 IMPLEMENTATION DETAILS

After having gone through different methods and techniques we finally choose RNN (specifically LSTM) as our model for hate speech detection. We will go through basics of both w.r.t. our project and then dive into implementation details of the same. Below images are taken from MIT6.S191 (introtodeeplearning.com).



Recurrent Neural Networks (RNNs)

Apply a **recurrence relation** at every time step to process a sequence:

$$\boxed{h_t} = \boxed{f_W}(\boxed{x_t}, \boxed{h_{t-1}})$$

Note: the same function and set of parameters are used at every time step

RNNs have a **state**, h_t, that is updated **at each time step** as a sequence is processed

Figure 2.8: RNN

In the above figure, a neuron is generalized over sequence of timestamps. With respect to our project, we have sequence of words input. Words at a particular index form a input at a fixed timestamp, and input to the neurons to that timestamp is also derived from output of the previous timestamps. Generally tanh activation function is used in Standard RNN. **The problem of Long Term dependency**: One issue with this model is exploding and vanishing gradient, i.e. during backward propagation if derivatives get very large we get exploding gradient problem, and if it gets very small we get vanishing gradient problem. During vanishing gradient, the update becomes negligible and the model remains inconclusive even after large number of iterations. To overcome this issue we use **Long Short Term Memory (LSTM)**.

2.3.1 LSTM

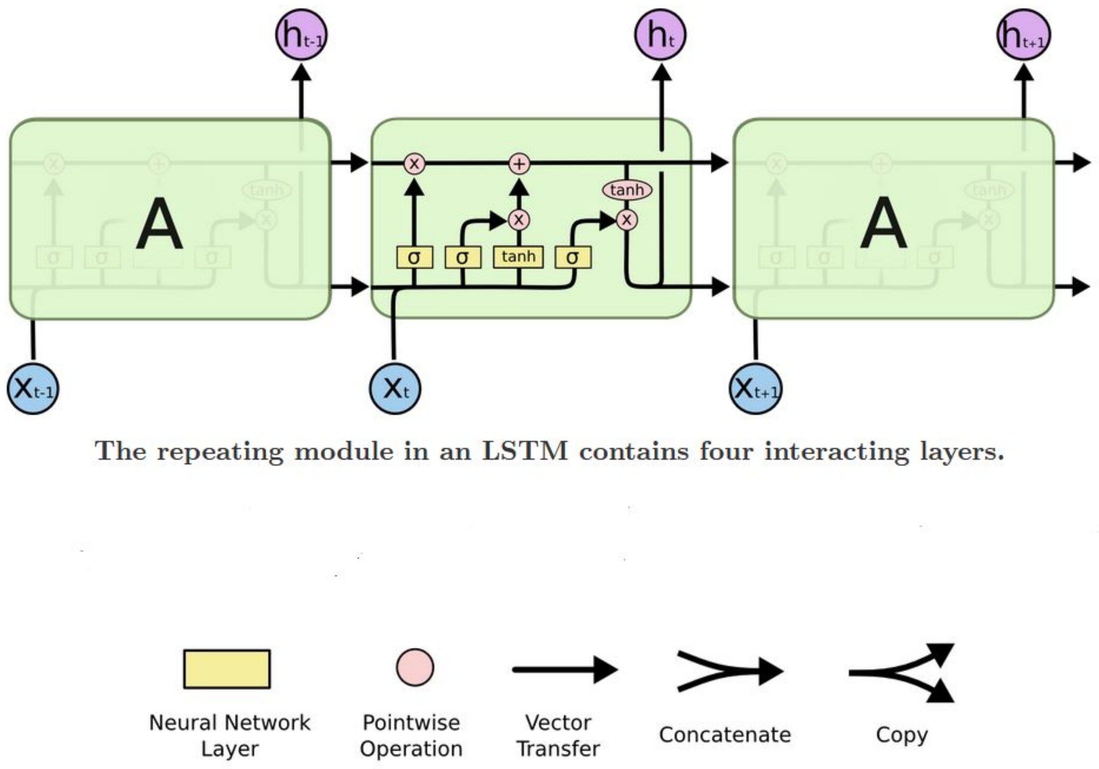
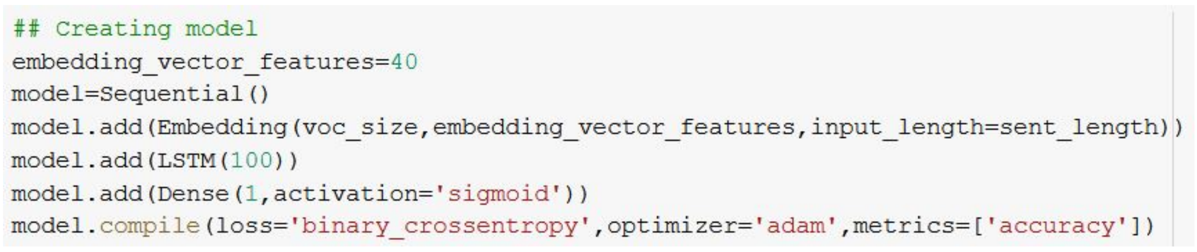


Figure 2.9: LSTM

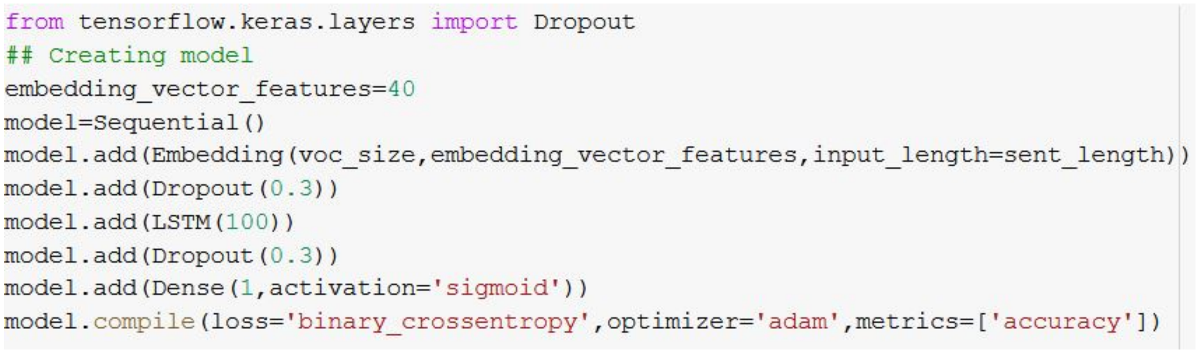
The horizontal line running across is the most important aspect of idea of LSTM. LSTM has the ability to include or remove some information to the cell state, using structures called gates. The figure 2.9 (taken from colah.github.io) it shows cell state along with the gates. Gates are the way to filter the information. They are made up of pointwise multiplication operation and a sigmoid neural net layer. The first gate in the cell is forget gate layer, and that uses the current input to decide what not to keep in the outpur from previous cell state. Sigmoid activation is used here which gives binary output to decide whether keep (1) or not (0). Then we have input gate layer, that uses current input to decide what has to be included in the memory state (line on the top of diagram). Then at last we have a gate that gives output using memory and current timestamp input.



```
## Creating model
embedding_vector_features=40
model=Sequential()
model.add(Embedding(voc_size,embedding_vector_features,input_length=sent_length))
model.add(LSTM(100))
model.add(Dense(1,activation='sigmoid'))
model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
```

Figure 2.10: LSTM Model

We will be using tensorflows keras for LSTM. One Hyperparameter which we can include is dropout, that improves the accuracy and runtime. We use dropout so that the model doesn't overfit by turning off some perceptrons so that dependency of that neuron reduces and others get the change.



```
from tensorflow.keras.layers import Dropout
## Creating model
embedding_vector_features=40
model=Sequential()
model.add(Embedding(voc_size,embedding_vector_features,input_length=sent_length))
model.add(Dropout(0.3))
model.add(LSTM(100))
model.add(Dropout(0.3))
model.add(Dense(1,activation='sigmoid'))
model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
```

Figure 2.11: Dropout

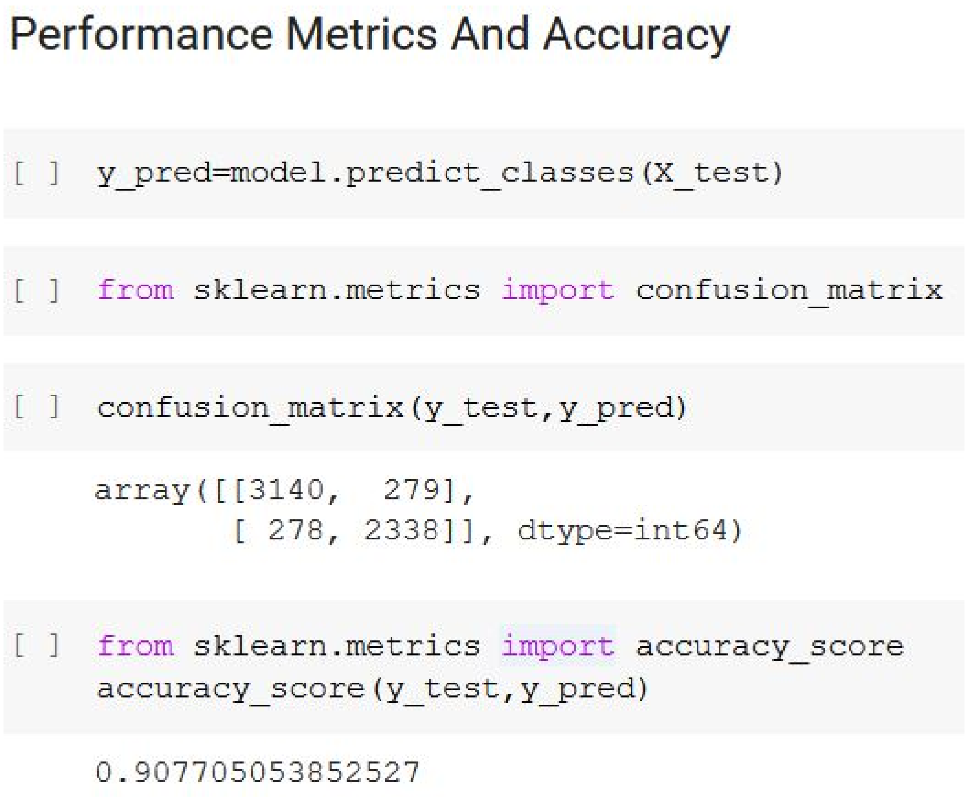
CHAPTER 3

RESULTS

Machine learning algorithms gave us maximum of 70% accuracy, then simple deep learning models gave us 80% accuracy. After having gone through so many different models we finally conclude that hate speech detection is best handled using sequential model, in our case that is Recurrent Neural Networks(RNN). We use special case of that called Long Short Term Memory (LSTM) that avoids the long term dependency issue of Standard RNN, and finally we add dropout for regularization to avoid overfitting of the model. Using these methods we are able to achieve an accuracy of near 91% for our dataset. Overall this is quite a practical approach towards hate speech detection. The model has used 10 iterations for training. There are still better accuracy chances if we can make use of CNN feature extraction ability and RNN sequence handling ability. One possible advancement could be usage of Bi-LSTM, but the improvement is negligible.

3.0.1 Future Work

A few research papers also suggest usage of CNN and Skipped CNN along with deep neural networks that can address the problem of Out-of-Vocabulary Words in word embeddings [5].


Performance Metrics And Accuracy

```
[ ]  y_pred=model.predict_classes(X_test)
```

```
[ ]  from sklearn.metrics import confusion_matrix
```

```
[ ]  confusion_matrix(y_test,y_pred)

     array([[3140,  279],
            [ 278, 2338]], dtype=int64)
```

```
[ ]  from sklearn.metrics import accuracy_score
     accuracy_score(y_test,y_pred)

     0.907705053852527
```

Figure 3.1: Performance Analysis

REFERENCES

[1] S. Malmasi and M. Zampieri, "Detecting hate speech in social media," in *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*, (Varna, Bulgaria), pp. 467–472, INCOMA Ltd., Sept. 2017.

[2] Y. Kim, "Convolutional neural networks for sentence classification," 2014.

[3] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, pp. 1735–1780, 11 1997.

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019.

[5] Z. Zhang and L. Luo, "Hate speech detection: A solved problem? the challenging case of long tail on twitter," *CoRR*, vol. abs/1803.03662, 2018.